Cooperative Visualization of Computational Fluid Dynamics

Michael J. Gerald-Yamasaki

Numerical Aerodynamic Simulation Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, California 94035-1000
USA

January 8,1993

Abstract

Tempus Fugit/Interview *is a computational fluid dynamics visualization application for which processing is distributed between high performance graphics workstations and supercomputers. Facilities are provided in the application for more than one user to view shared images creating a cooperative visualization environment. The way in which the computation is partitioned between the supercomputer and the workstations is critical to the capability of the application to present simultaneous, identical, animated images of fluid dynamics to more than one user.*

Keywords: Scientific Visualization, Computer Supported Cooperative Work, Distributed Graphics

1. Introduction

The increased capabilities of supercomputers have been used to dramatically increase the size and complexity of numerical simulations of fluid flow [16]. As the size of the simulations increase, the size of the flow solution data also increases and can result in immense data sets representing the physical characteristics of a flow field. The greatest impediment to developing systems for visualization of three-dimensional, unsteady fluid flow is the size of the data, which constrains interactive response time.

As applications utilize the advanced features of high-performance graphics workstations to visualize complex time-dependant data, the individual scientist is presented with more and more information-laden images. Animated images created by such visualization applications are not transportable beyond the workstation without some loss of informational content. Consequently, sharing the information acquired during the visual analysis process among collaborating scientists is difficult.

The visualization of unsteady fluid flow presents two connected problems: how to provide interactive exploration capability over extremely large data sets and how to make the exploration a process which can be shared with another scientist.

This paper examines the architecture of *Tempus Fugit* ("time flies"), a tool for visualizing unsteady fluid flow. Processing in Tempus Fugit is distributed between a high performance graphics workstation and a supercomputer. Images of fluid flow characteristics are animated over the time dimension of the time-dependant, unsteady fluid flow solution data. The companion program, *Interview*, provides facilities to share the supercomputer computational environment with a second workstation, creating a cooperative visualization environment.

The first part of the paper will describe the requirements imposed by shared visualization of large time-dependent data sets in a distributed environment of supercomputers and graphics workstations. The second part of the paper will analyze the architecture of computer conferencing applications and distributed visualization applications. The next part of the paper will analyze the architecture and implementation of Tempus Fugit/Interview. Finally, concluding remarks on architectural choices end the paper.

2. Background

Despite advances in the delivery of computational power to users of high-performance graphics workstations, there remain visualization applications for which the computational requirements can only be met by supercomputers. Distributed processing is used to provide the user with the combined capabilities of a high performance graphics workstation and a supercomputer under the control of a single application [7, 20]. The supercomputer's capabilities of great processing power, large memory, large disk storage, and fast disk access are necessary to contain, access, and calculate over, the large data sets endemic to unsteady fluid flow analysis. The high speed graphics of the workstation transform numerically calculated data into high resolution animated images of fluid flow features. The human-machine interface, also provided by the workstations, adds mechanisms for controlling the analytical tools of the visualization system.

The high resolution, three-dimensional, color, animated images which are the result of the visualization process are not transportable beyond the workstation without some loss of informational content. Much of the three-dimensional character of the images is lost without the capability to perform interactive view transformations. Recording the animated images using video is limited by the resolution of NTSC video encoding and eliminates interactive capability. Color printing or photography is comparable in color resolution and in some cases is superior in image resolution to the workstation graphics monitor. However, much of the information to be analyzed is in the animation of the images and cannot be captured with still images.

The lack of transportability of these images makes it difficult to communicate the results of the visualization process to collaborators, which in turn makes the collaboration process that much more difficult. A cooperative visualization tool, which would allow users on separate workstations to simultaneously view the images as they are created, would create a shared environment for analyzing the images and facilitate the dialog so important to collaboration.

Recently developed computer tools which facilitate collaboration have shown that a WYSIWIS ("what you see is what I see") interface is valuable. This type of interface provides for the "presentation of consistent images of shared information to all participants" [25, 26]. Such an interface to a visualization application would allow scientists to see and interact with each other's work through their workstations. Visualization, then, becomes a communication medium and the graphics workstation a platform for the exchange of ideas.

One model of a collaborative environment is represented in the simple diagram in figure 1 [22]. While conversation is serial and ephemeral in nature, shared space is substantial and provides another medium for communication.

Figure 1: Collaborative Environment Model.

Shared access to information makes symbolic representation more concrete. Stefik, *et al.* use the chalkboard as a metaphor for a shared space for information storage [26]. The idea of WYSIWIS follows somewhat from this basic model.

With typical visualization applications a single user is "alone" with the data and the images which are the result of the visualization process. When an interesting image is produced on the graphics monitor, it's common in our laboratory to call co-workers to the monitor to see what has been produced. With the shared view of the monitor, the collaborative environment outlined above is created (figure 2). Collaborators who are in another building or another city however, cannot participate in this interaction.

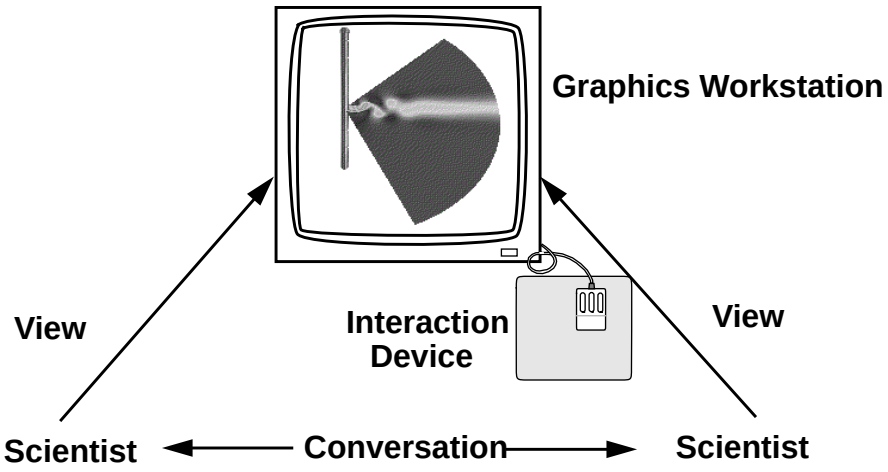

Figure 2: "Gather Around" Collaborative Environment.

Tempus Fugit/Interview builds on the basic model to provide an environment where distance is not a deterrent to creating the collaborative environment. For Tempus Fugit/Interview, the shared space resides on the supercomputer and the images are rendered on separate graphics workstations (figure 3).

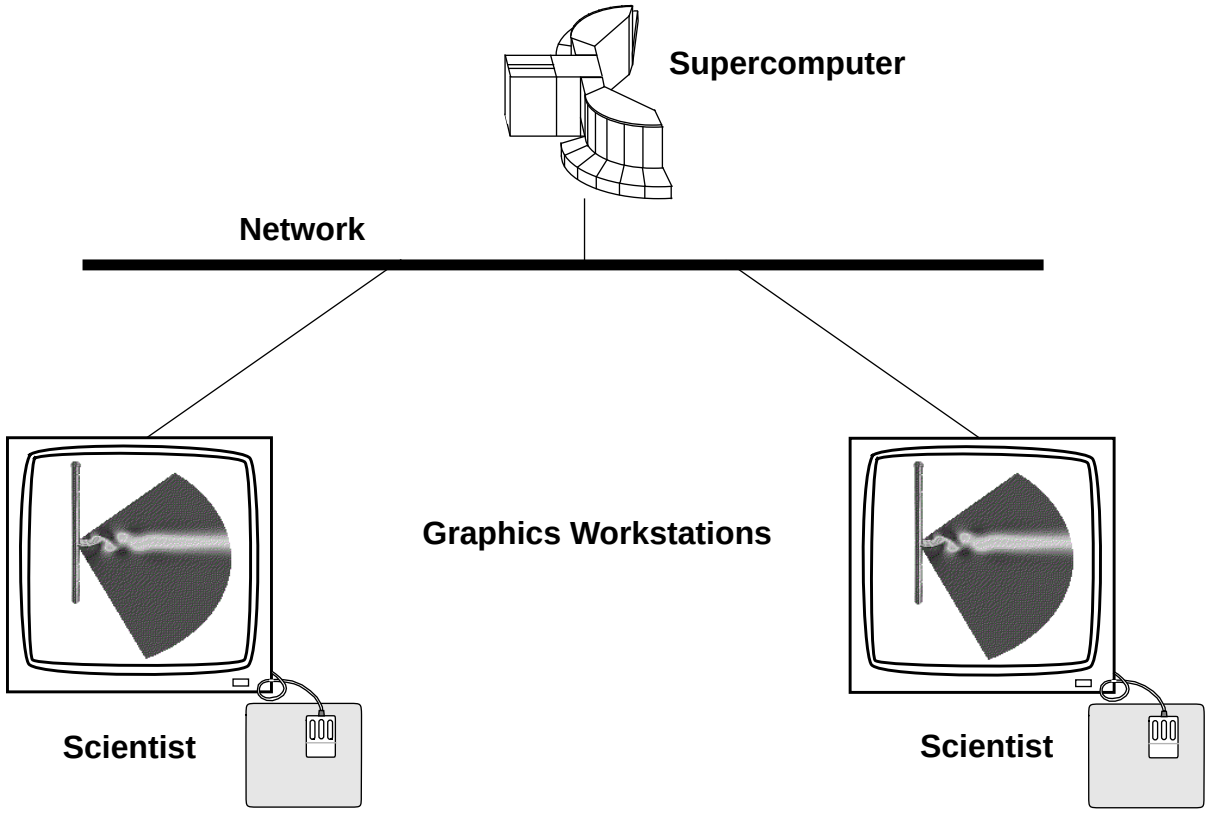

Figure 3: Tempus Fugit/Interview Collaborative Environment.

3. The Computational Fluid Dynamics Application

One of the main objectives of the Numerical Aerodynamic Simulation (NAS) Program at the National Aeronautics and Space Administration (NASA) Ames Research Center is the provision of a comprehensive computing environment to facilitate computational aerodynamics and fluid dynamics research [2]. To this end, the NAS Processing System Network (NPSN) was developed. The NPSN contains a wide range of computer systems, including several supercomputers (Convex C3240, Cray 2 4/256, Cray YMP 8/256, Intel IPSC 860, TMC 32K CM2) and an array of Silicon Graphics IRIS workstations. Several networks provide connectivity and a basis for network development and research. These networks include Ethernet, UltraNet, and FDDI.

Recent computational fluid dynamics (CFD) research at NAS has produced data sets for which interactive visualization is, due to the data size, beyond the capabilities of currently available visualization packages. Visualization of data sets of the magnitude of those described in Table 1 stress the computational resources of even the best equipped computer centers. It has been suggested that visualization of these large data sets involves tradeoffs among time, space and flexibility constraints [13].

Data Set	Zones	Total Nodes	Solution Size (MB) Per Time Step	Saved Time Steps	Total Size (GB)
STOL w/ Thrust Reversers [6]	4	983,366	18.76	9000	168.8
SOFIA Airborne Observatory [1]	35	3,314,575	63.22	100	6.3
Complete STOVL Aircraft [24]	18	2,833,700	54.05	100	5.4
Tapered Cylinder [15]	1	131,072	2.50	800	2.0

Table 1: Representative Unsteady Fluid Flow Data Sets

CFD research can be divided into three steps: grid generation, numerical simulation, and post-process analysis. A numerical grid is created describing an object and the surrounding space. Flow solvers calculate physical properties of the flow at the nodes of the grid.

A typical data set consists of a grid file and one or more solution files. Typical grids consist of one or more zones constructed over a curvilinear coordinate system. These zones may overlap. The grid file contains the x-, y-, and z-coordinate values of the grid nodes mapping the nodes to Cartesian space. Solution files contain the values for density, energy, and momentum for each grid node. Density and energy are scalar values while momentum is a three-dimensional vector. A steady state solution data set contains a grid file and a solution file. An unsteady solution data

set contains a grid file and a solution file per time step.

An example of a multiple zone grid is shown in figure 4 for a complete short take-off and vertical landing (STOVL) aircraft [24]. Figure 4A shows a wire frame of a single zone of the grid with cross hairs on the surface nodes. Figure 4B shows crosshairs for all of the exterior nodes of the zone to illustrate the grid density and the curvilinear nature of the grid layout. The zone consists of 75,330 nodes, while the entire grid consists of 2,833,700 nodes. Figure 4C wire frames for the zones which contain the aircraft surface and finally, figure 4D shows all eighteen zones in wire frame.

From the density and energy scalar fields and momentum vector field, other scalar and vector fields can be calculated. The widely used PLOT3D CFD visualization tool developed by Pieter Buning [33] provides facilities for building fields for about one hundred different scalar and vector functions and provides the model for many CFD visualization tools. The FAST CFD visualization tool provides a scalar and vector field calculator as well as a number of built-in functions [3].

A variety of visualization techniques can be applied to these scalar and vector fields for post-process analysis. These include particle paths, isoscalar surfaces, cutting planes, and topology. Graphics techniques are applied to color, shade, light, project and otherwise render images on the workstation monitor.

4. Architecture

4.1 Computer Conferencing Architecture

Systems which provide the infrastructure for developing computer conferencing applications usually make the choice between a centralized or a replicated architecture (figure 5) [8, 17, 19]. This choice is predicated on a desire to require minimal modification to single user applications when operated in a computer conference or cooperative environment.

A single instance of the application is used in the centralized architecture with a conference manager which coordinates the I/O between the clients and the application. An instance of the application per client is used in the replicated architecture. The conference manager in the replicated architecture is interposed between the client and the application in the input stream.

A major issue in the difference between these two architectures is the ability to provide both shared and private contexts while maintaining consistent state. The centralized architecture is simpler but unable to provide the capability of maintaining both shared and private contexts without modification of the underlying application. The replicated architecture is a flexible architecture but presents the difficulty of maintaining consistent state across all replicants. In both architectures

the conference manager implements the distribution of processing over the network.

Computer conferencing architectures involve distributed processing to distribute locality. The output of an application is distributed to more than one location. Distributed processing can also be used to acquire resources that are not locally available, such as the acquisition of supercomputer resources from a workstation. A distributed visualization application is an example of this latter use of distributed processing.

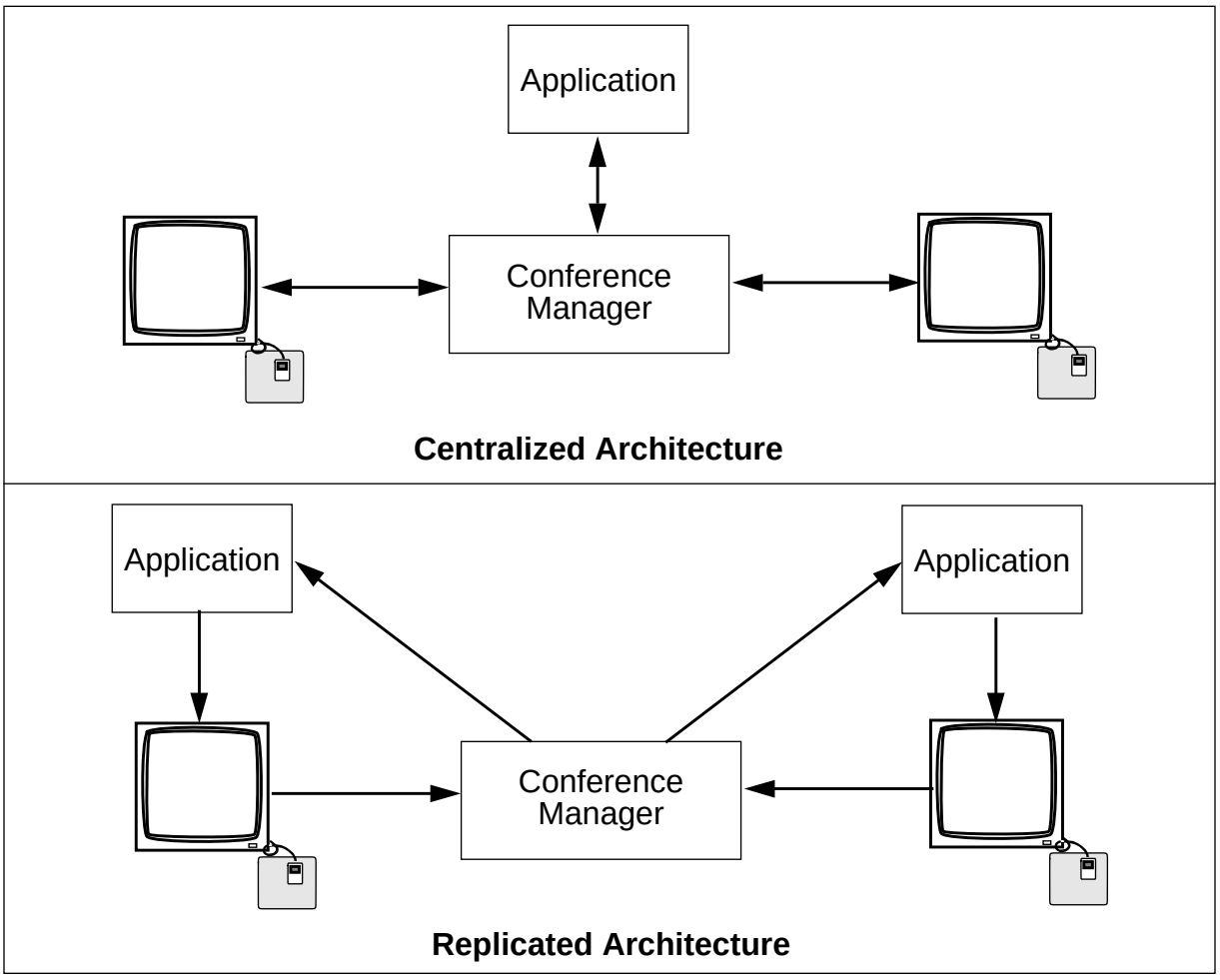

Figure 5: Computer Conferencing Architectures.

4.2 Distributed Visualization Application Architecture

The CFD visualization process can be understood as the transformation of data (figure 6). Visualization techniques are applied to the scalar and vector fields, which make up the raw data, to form geometry data. Geometry data is rendered to form image data.

Distributed processing can be applied to the visualization process by partitioning the computational tasks between a supercomputer and high performance graphics workstations. How this partitioning is accomplished will determine the ability of the

visualization system to provide an environment which best utilizes the capabilities of the graphics workstations, the supercomputer, and the network.

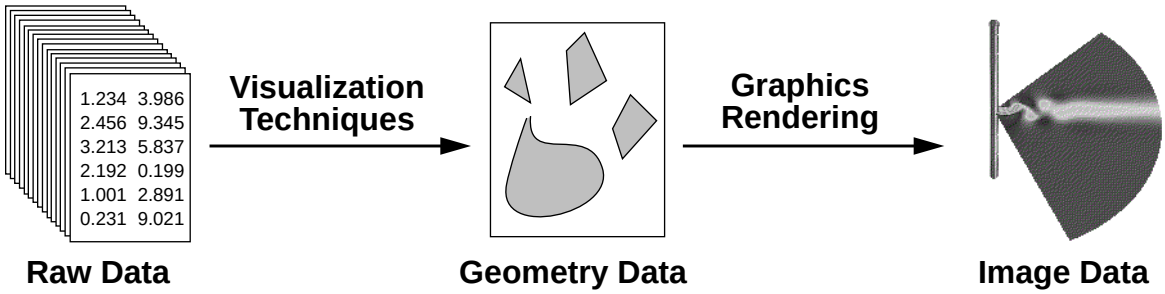

Figure 6: Visualization Process.

Figure 7 illustrates several possible partitions: *the distributed file system partition, the geometry partition, the distributed graphics library partition, and frame buffer partition*. Looking at the visualization process as a whole, the distributed file partition imports the input from a remote machine, while the frame buffer partition exports the output to a remote machine. The distributed graphics library and geometry partitions separate local and remote tasks at the point of graphics processing.

The use of distributed file systems such as Network File System (NFS) [21] is often given as an example of distributed processing. For a visualization application, the raw data resides on the server machine and is made available for graphics processing on the workstation client. Using distributed processing in this manner allows the client to utilize the disk resources of the server. Another example of this type of partition is when a server process creates the raw data and delivers it directly to a client process. In CFD visualization this method has been used to preview data as it is produced by a flow solver. Intermediate solutions are produced by a flow solver and transferred over the network to a visualization application instead of being written to disk.

A small advantage in saving disk access time and space is gained from this distributed file system partition. However, for a visualization application this partition of processing would only utilize the high disk-to-memory speed of the supercomputer, leaving the computationally intensive processing of the data to the workstation. To gain the benefit available by distributing processing it is essential that the transformation of raw data to geometry data be carried out on the supercomputer.

For the frame buffer partition, all of the processing is accomplished on one machine and the image is transferred to another for display. The image can be transferred as pixel image data or further transformed into video [12,14].

This has been found to be useful in that the image display takes place at a different machine and location than the calculation and rendering of the image, diminishing a

requirement of geographical proximity of the user to the machine which is carrying out the calculation.

Transfer of images can turn out to be quite a large amount of data to transfer over the network, perhaps 24 bits per pixel by 1024 by 1280 pixels per frame. The data transfer rate, at 24 frames per second, would then be approximately 90 MBytes per second. Maintaining this transfer rate to two or more workstations or frame buffers in a cooperative effort would be difficult. Using video transmission can decrease the volume of data to be transferred but only at the cost of greatly reducing the quality of the images.

While the advent of high speed networks in the gigabit-per-second range [5, 11] removes a potential bottleneck in the performance of these distributed applications, high speed networks are not a panacea. Even though some networks may be capable of transferring data at a rate of one gigabit-per-second, it will be some time before workstations are capable of transferring or receiving data at that speed. Even so, image data transfer at animation speeds would be difficult to sustain.

The distributed file system partition and the frame buffer partition allow the entire visualization process (figure 6) to be accomplished on one machine with either the raw data being imported from a remote machine or the output exported to a remote machine. These two partitions require minimal modification to applications implemented to operate only in a non-distributed environment to allow the applications to operate in a distributed environment. The tradeoff for ease of implementation is a limitation in the flexibility to utilize the combined processing power of the supercomputer and graphics workstation.

A third distributed processing partition is at the transformation of geometry data to image data found in the distributed graphics library partition. The raw data is processed and formatted to form geometry data. The graphics routines transform the geometry data into images. The sequence of graphics library calls may themselves be transferred over the network and executed on another machine. This partitioning has the advantage that the programming effort is carried out on the computation machine (supercomputer) and graphics calls made as though the image creation was local even though it its carried out on another machine. A special distributed graphics library [23] which implements all of the necessary network transactions is used. Network window systems are also an example of the utilization of the distributed graphics library type of partition [18].

The distributed graphics library partition is designed to have minimal impact on the basic design of an application which is being modified to distribute processing. This partition also has limited flexibility. While the use of a distributed graphics library partitions the computation in a way which utilizes the strengths of the supercomputer and of the graphics workstation, it is often advantageous to be able to use distributed processing which does not involve graphics. This ability is unfortunately not available with the distributed graphics library approach.

Interactions which control the production of images from the geometry data require network transactions with a distributed graphics library partition. The network transactions and the associated processing overhead can be reduced by moving the geometry data to the workstation. With this geometry partition, the entire rendering process can then be completed on the workstation without involving the network for transferring command information or data. The geometry partition is a very flexible framework for distributed visualization applications. Unlike the distributed graphics library partition, the geometry partition separates network processing from graphics processing. Herein lies the flexibility to carry out distributed processing functions without involving graphics.

The geometry partition allows the rendering controls to the visualization process to be completely local to the graphics workstation. If the geometry data for an entire animation can be contained on the workstation, the entire viewing process of an animation including view transformations, color manipulation and other rendering controls is completely local to the workstation and does not incur any network transaction overhead.

4.3 Combined Architecture

To combine the ability of the distributed visualization application to distribute compute intensive and memory consumptive operations to a supercomputer with a computer conferencing architecture requires some architectural modification. To adopt a centralized architecture would perhaps be the most straight forward as the conference manager could continue to manage all network traffic. The inability of providing both shared and private contexts remains a problem with this architecture. The replicated architecture would require network transactions to be somehow interposed in the output stream as well as the input stream to allow the application to be operated on the supercomputer.

In order to combine the flexibility of the geometry partition with a computer conferencing architecture a hybrid of the centralized and replicated architectures was developed (figure 8). The computationally intensive transformation of raw data into geometry data is centralized into a process on the supercomputer. The transformation of geometry data to images is replicated on the graphics workstations. This hybrid architecture allows for both shared and private contexts for rendering control since that portion of the application is replicated. Maintenance of consistent state is simplified by the centralized portion of the application, which controls the transformation of raw data to geometry data using visualization techniques.

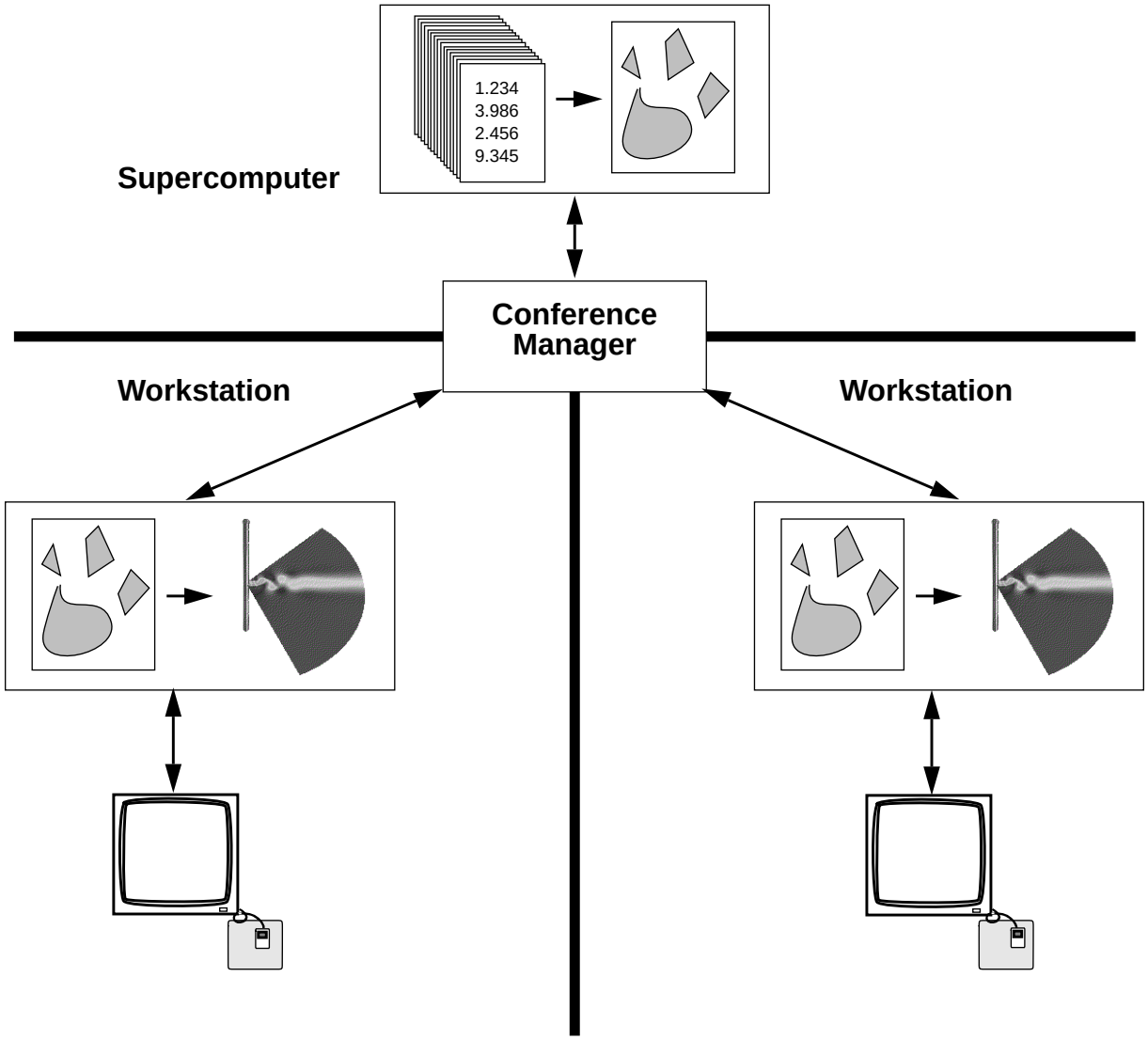

Figure 8: Geometry Partition with Hybrid Conferencing Architecture

5. Implementation

Tempus Fugit is a stand-alone application for visualizing unsteady fluid flow which uses a geometry partition to distribute processing. Interview is a companion program to Tempus Fugit which replicates the geometry data to image data transformation on a second graphics workstation.

5.1 Distribution of Computation

The main vehicle for distributing computation between supercomputers and

workstations in Tempus Fugit/Interview is Distributed Library (dlib) [30]. Like many systems which provide for distributed processing, dlib is based on the remote procedure call (RPC) model [4, 10, 27, 29]. However, unlike most of these systems, dlib was developed to provide a service which allows for a conversation of arbitrary length within a single context between client and server. The dlib server process is designed to be capable of storing state information which persists from call to call, as well as allocating memory for data storage and manipulation. While RPC protocols are frequently likened to local procedure calls without side effects, dlib more closely resembles the extension of the process environment to include the server process.

Dlib provides utilities for automatic stub generation to handle the transfer of the information necessary to execute the routine in the remote environment. Due to the persistent nature of the remote environment, dlib is able to coordinate allocation and use of remote memory segments and provide access to remote system utilities.

Dlib was originally designed on a model of one client to one server. To allow multiple clients to share the server process environment, the dlib server was modified to accept more than one connection. Each connection is selected for service by the server process in the sequence that the dlib calls are received. The dlib calls are executed by the server in a single process environment as though there were only one client. With this modification, dlib becomes the conference manager for the centralized resource which executes on the supercomputer. An additional communication path between the two clients allows the input streams to be shared for the replicated portions of the application.

5.2 Tempus Fugit

Tempus Fugit is a stand-alone application for visualizing unsteady fluid flow. The transformation of raw data to geometry data by the application of various visualization techniques is accomplished on the supercomputer and controlled by a mouse-driven interface on the workstation. The geometry data is transferred over a connection to the dlib client on the workstation where time-sequenced animated images are produced as in the geometry partition described above. This dlib client will be referred to as the Tempus Fugit client.

The process of selecting a visualization technique to be applied, transforming raw data to geometry data, transferring the geometry data to the workstation, and rendering the data into an animated image can result in an animation containing a number of visual elements. Figure 9 shows several *grid surface*s, two dimensional slices through the curvilinear grid, and several *isosurfaces* over subranges of the grid, surfaces of constant scalar value, for the velocity magnitude of flow about a tapered cylinder [15]. The series of frames at the bottom of the figure gives an idea of the animation of the scene.

5.3 WYSIWIS

The Tempus Fugit client will have been active for an arbitrary length of time when Interview is invoked. As such, the image the Tempus Fugit client is presenting may contain a number of visual elements. Interview creates a connection to the dlib server on the supercomputer and to the Tempus Fugit client. The Tempus Fugit client maintains a list of descriptions of the visual elements it is currently viewing. Upon request this list is sent to the Interview client. From this list, the Interview client has the information to transfer the geometry data from the server process using dlib calls in the same way that the Tempus Fugit client received the data.

The Interview client is now able to present images created from the same geometry data as the Tempus Fugit client. Interactive controls for view transformations and animation sequencing are individual to each client. Consequently, the two clients at this point are viewing the same three-dimensional geometry data but may have different viewing perspectives.

The ability to individually view the same geometry data is analogous to two people looking at a three-dimensional object, say, an open book. One person can see the title and author on the front cover, while the other can read the pages. While their views are different, there is a shared context for conversing.

To present an identical image on both monitors simultaneously, rendering controls such as view transformations and animation sequencing must be consistent. The graphics transformation matrix controls the mapping between geometry data and the screen representation. In order for identical images to be viewed by both clients, the rendering control information of one client is sent to the other client. The receiving client loads the transformation matrix into its graphics pipeline creating an image identical with the viewing perspective of the sending client. The animation sequencing information is used to synchronize the animation frame by frame.

Each client has a set of mouse-driven view transformation controls for rotation, pan, and zoom. To see what is being presented by the other client “tracking mode” is selected. This results in a message sent to the other client to begin sending the rendering control information. The other client continuously updates the rendering control information until tracking mode is deselected and “detached mode” is entered. While in tracking mode, view transformation controls are disabled. While in detached mode, the client is able to control the viewing perspective (Figure 10).

When the geometry data is transferred to two workstations, the basis for sharing the information has been established. The rendering portion of the application is replicated by the two clients. With a copy of the geometry data, each workstation can render images distinct from the other workstation in a private context. By exchanging the rendering control information the two workstations can generate identical images in a shared context. There is in a sense two levels of sharing which can be implemented with the hybrid conferencing architecture: sharing the geometry

data and sharing rendering controls such as view transformations.

5.4 Current Status and Future Work

A prototype has been implemented which targeted single zone data sets such as the tapered cylinder [15]. The raw data for this prototype resided on the supercomputer's disk. The transformation of raw data to geometry data included extracting portions of the raw data from disk. The efficiency of the system was highly dependent on disk performance and varied greatly depending on distribution of the required data on disk.

Unfortunately, a disk distribution optimization for one visualization technique conflicts with an optimization for another visualization technique. In the prototype, sample data is organized to have time dimensions of data arrays ordered sequentially on disk. This optimizes disk reads over spatial subsets for such visualization techniques as grid surfaces, cutting planes, and isosurfaces over subsets of the grid. Integral curve visualization techniques perform best with a disk organization with the spatial dimensions of data arrays ordered sequentially. Additional work is required to find an organization for disk-resident data which is effective for both types of visualization techniques. Management of the data through the hierarchy of data storage devices was identified as a major issue effecting efficiency and interactive performance.

The prototype was sucessful in demonstrating the hybrid conferencing architecture could provide an efficient framework for sharing the interactive exploration of extremely large data sets. Tracking mode was found to be especially effective with this architecture. The transformation matrix and other rendering control information are very compact and can be transferred between workstations well within the animation frame rate of between ten and thirty frames per second on networks of modest speed such as Ethernet. By replication of the process required to transform geometry data into images, only the rendering control information need be transferred between the workstations to provide the WYSIWIS function.

A second prototype is currently being developed which targets multi-zone data sets. It has been observed that for many of these large data sets the area which contains the interesting features of the flow is concentrated in a small percentage of the total gridded volume. A flexible subsetting tool has been developed which interactively extracts spatial and temporal subsets, controlling the size of the subset to fit in available supercomputer memory. Using this approach allows the visualization techniques to be applied to the raw data without accessing the disk and acquiring the concomitant delay. It is hoped that the tool will concentrate the speed mismatch between disk access and interactive visualization to a single set-up operation without diminution of exploratory capability.

6. Conclusion

The way in which computation is partitioned between constituent processors is an important design consideration for distributed and cooperative applications. Many applications have been implemented for use on a single machine before it becomes desirable to operate the application in a distributed and/or cooperative environment. For an application such as the visualization of CFD, processing can be partitioned in ways which require minimal modification to a single processor implementation. However, such partitions are limited in their ability to take advantage of the full range of facilities available in a distributed environment.

The advocacy for either a centralized or replicated architecture for computer conferencing applications is predicated on a desire to require minimal modification to single-user applications. While these architectures are effective for distributing locality, it is difficult to utilize these architectures in a way which also acquires the computational advantages available through distributed processing. Efficiencies in the passing of control, in providing shared and private contexts, and in providing computational services, should guide which elements should be a centralized resource and which should be replicated. The hybrid architecture presented above provides the flexibility to utilize distributed processing to garner the computational resources required to interactively explore extremely large data sets while providing the ability to share the exploration process.

7. Acknowledgments

The author would like to thank E. Lisette Gerald-Yamasaki and Tom Lasinski for their continuing support.

8. References

[1] Atwood, C. A. and Van Dalsem, W. R. Flowfield simulation about the SOFIA airbourne observatory. 30th American Institute of Aeronautics and Astronautics (AIAA) Aerospace Sciences Meeting and Exhibit (Reno, NV. Jan. 1992) AIAA Paper 92-0656.

[2] Bailey, F. R. Status and projections of the NAS program. In C*omputational Mechanics - Advances and Trends.* A. K. Noor, editor New York: American Society of Mechanical Engineers, 1986, 7-21.

[3] Bancroft, G. V., Merritt, F. J., Plessel, T. C., Kelaita, P. G., McCabe, R. K. and Globus, A. FAST: a multi-processed environment for visualization of computational fluid dynamics. In *Proc. Visualization '90* (San Francisco, CA, Oct. 23-26, 1990) 14-23.

[4] Birrell, A. D., and Nelson, B. J. Implementing remote procedure calls. *ACM Trans. on Comp. Sys.* 2, 1 (Jan. 1984), 39-59.

[5] Chlamtac, I. and Franta, W. R. Rationale, directions, and issues surrounding high speed networks. 78, 1 (Jan. 1990), 94-120.

[6] Chawla, K. and Van Dalsem, W. R. Numerical simulation of STOL operations using thrust reversers. AIAA Aircraft Design Systems Meeting (Hilton Head, S.C. Aug. 24-26, 1992) AIAA Paper 92-4254.

[7] Choi, D. and Levit, C. Implementation of a distributed graphics system. *Internat. J. Supercomput. Appl.* (Winter 1987), 82-95.

[8] Crowley, T., Milazzo, P., Baker, E., Forsdick, H. and Tomlinson, R. MMconf: an infrastructure for building shared multimedia applications. In *Proc. CSCW '90* (Los Angeles, CA, Oct. 7-10, 1990) New York, NY: ACM (Order No: 612900), 329-342,

[9] Dewan, P. and Choudhary, R. Flexible user interface coupling in a collaborative system. In Proc. CHI '91 (New Orleans, LA, Apr. 27-May 2, 1991) New York, NY: ACM (Order No: 608910), 41-48.

[10] Dineen, T. H., Leach, P. J., Mishkin, N. W., Pato, J. N., and Wyatt, G. L. The network computing architecture and system: an environment for developing distributed applications. *Proceedings of Summer Usenix* (June 1987), 385-398.

[11] Farber, D. Gigabit network testbeds. *Computer* 23, 9 (Sep. 1990), 77-79.

[12] Fowler, Jr., J. D., and McGowen, M. Design and implementation of a supercomputer frame buffer system. In *Proc. Supercomputing '88* (Orlando, Florida, Nov. 14-18, 1988) Washington, DC: IEEE Computer Society Press (Order No. 882), 140-147.

[13] Globus, A. A software model for visualization of time dependent 3-d computational fluid dynamics results. *NAS Applied Research Technical Report* RNR-92-031 (Nov. 1992). For *NAS Applied Research Technical Reports* send e-mail addressed to doc-center@nas.nasa.gov or telephone (415) 604-4632.

[14] Haber, R. B., and McNabb, D. A. Eliminating distance in scientific computing: an experiment in televisualization. *Internat. J. Supercomput. Appl.* 4, 4, (Winter, 1990), 71-89.

[15] Jespersen, D. and Levit, C. Numerical simulation of flow past a tapered cylinder. American Institute of Aeronautics and Astronautics (AIAA) paper 91-0751. AIAA 29th Aerospace Sciences Meeting. (Reno, Nevada, Jan. 7-10,

1991).

[16] Lasinski, T., Buning, P., Choi, D., Rogers, S., Bancroft, G. and Merritt, F. Flow visualization of CFD using graphics workstations. *Proc. AIAA 8th Computaional Fluid Dynamics Conf.* (Honolulu, Hawaii, June 9-11, 1987) AIAA Paper 87-1180, 814-820.

[17] Lauwers, J. C. and Lantz, K. A. Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems. In Proc. CHI '90 (Seattle, WA, Apr. 1-5, 1990) New York, NY: ACM (Order No: 608900), 303-311.

[18] O'Reilly & Associates Inc. *X Window System Series* Sebastapol: O'Reilly, 1990.

[19] Patterson, J. F., Hill, R. D., Rohall, S. L. and Meeks, W. S. Rendezvous: an architecture for synchronous multi-user applications. In *Proc. CSCW '90* (Los Angeles, CA, Oct. 7-10, 1990) New York, NY: ACM (Order No: 612900), 317-328,

[20] Rogers, S. E., Buning, P. G., and Merrit, F. J. Distributed interactive graphics applications in computational fluid dynamics. *Internat. J. Supercomput. Appl.* 1, 4 (Winter 1987), 96-105.

[21] Sandberg, R. The Sun Network File System: design, implementation, and experience. Sun Microsystems, Inc. (1986).

[22] Schrage, M. *Shared Minds.* New York: Random House, 1990

[23] Silicon Graphics Computer Systems. Using the distributed graphics library. *4-Sight Programmer's Guide* Document Number 007-2001-030 (1990).

[24] Smith, M., Chawla, K., and Van Dalsem, W. Numerical simulation of a complete STOVL aircraft in ground effect. AIAA 9th Applied Aerodynamics Conference (Baltimore, MD Sept. 23-25, 1991) AIAA Paper 91-3293.

[25] Stefik, M., Bowbrow, D. G., Foster, G., Lanning, S. and Tatar, D. WYWIWIS revised: early experiences with multiuser interfaces. ACM Trans. on Office Info. Sys. 5, 2 (Apr 1987), 147-167.

[26] Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Commun. ACM* 30, 1 (Jan. 1987), 32-47.

[27] Sun Microsystems. *Request for Comment #1057* Network Working Group (June, 1988).

[28] Walatka, P. P. and Buning, P. G. Plot3d user's manual. NASA Technical Memorandum 101067, NASA Ames Research Center.

[29] Xerox Corporation. Courier: the remote procedure call protocol. *Xerox System Integration Standard (XSIS) 038112*, (Dec. 1981).

[30] Yamasaki, M. Distributed library. *NAS Applied Research Technical Report* RNR-90-008 (Apr. 1990).